Propositional Logic (Rosen, 6th edition, Sections 1.1 - 1.2) Carol Zander

Logic is extremely important. Every time you use an if-then statement, you are using logic to determine the outcome of your program. However, it goes much deeper than this. Logic is the study of reasoning - of determining whether statements are true or false. If you don't have a good foundation in logic and logical reasoning, then it is difficult to reason effectively about your programs. Propositions deal with statements that are either true or false.

Proposition -- any statement that has one truth value, true or false.

They do not need to be true to reason about them. The following are propositions:

Starbucks sells coffee in Seattle.
Metallica plays easy listening music.
Brandi Carlile will win a grammy in 2012. (True or false, we just don't know which yet)

Are these statements true or false?
Questions and commands are not propositions:

Do I own any cats?
Finish your programming assignment.

When dealing with logic abstractly, we won't pay much attention to what the particular propositions mean. We will give them abbreviated names (lower-case letters) to represent them and reason about them using their abbreviations. So, "p" could represent the proposition
"My car is white" and "q" could represent the proposition "My car is a Ferrari".

Let p, q represent propositions, p and q are propositional variables.

Propositions which are logical expressions use logical operators such as AND, OR (inclusive OR, one or both), XOR (exclusive OR, one, but not both), NOT. Many different symbols typically represent these operators.

Recursive definition of logical expressions:
Basis -- Propositional variables, logical constants (true, false) are logical expressions.

Recursive -- If p and q are logical expressions, then so are p AND q, p OR q, NOT p, etc.

We can negate statements, i.e., change their truth value.
For example, NOT p is "My car is not white" .
If r is "n < 2", then NOT r is "n is not less than 2" , or "n >= 2" .
If p is "Today is Tuesday" and q is "It is raining" then p OR q is "Today is Tuesday or it is raining" .
p and q is "Today is Tuesday and it is raining" .

Operator precedence

		C++	
high			
|	NOT	!	negation
|	AND	&&	conjunction
|	OR	||	disjunction
|	→		implication, if-then
|	↔		equivalence, biconditional, if and only if (iff)
v			
low			

For any particular operator, we can define it using a truth table that says what the value is for any possible combination of inputs. The following tables show results of basic logical operations.
A zero represents false, one represents true.

p	q	p AND q	p OR q	NOT p
0	0	0	0	1
0	1	0	1	
1	0	0	1	0
1	1	1	1	

p	q	$p \rightarrow q$ which is same as (NOT p OR q)
0	0	1
0	1	1
1	0	0
1	1	1

p	q	NOT p	NOT p OR q
0	0	1	1
0	1	1	1
1	0	0	0
1	1	0	1

p	q	$p \leftrightarrow q$ which is same as $(p \rightarrow q$ AND $q \rightarrow p)$
0	0	1
0	1	0
1	0	0
1	1	1

p	q	$p \rightarrow q$	$q \rightarrow p$	$(p \rightarrow q$ AND $q \rightarrow p)$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	1	1	1

p	q	$p \oplus q$ exclusive OR
0	0	0
0	1	1
1	0	1
1	1	0

Logical expressions are equivalent if they have the same truth values under all conditions. You can prove expressions are equivalent using truth tables.

Tautology -- logical expression whose value is true under all situations, E.g., `true OR p`
Contradiction -- logical expression whose value is false under all situations, E.g., false AND p

Consider $p \rightarrow q$, variations on this logical expression include
- contrapositive: NOT $q \rightarrow$ NOT p (equivalent to $p \rightarrow q$)
- inverse: NOT $p \rightarrow$ NOT q
- converse: $q \rightarrow p$

In section 1.2, Table 6 (pg. 24), there are many logical equivalences.
Some are clear such as the commutative laws: (p OR q) is equivalent to (q OR p).
Two useful laws are DeMorgans laws.

DeMorgans laws:
- complement of the product is the sum of the complements
- complement of the sum is the product of the complements

NOT (p AND q) $\leftrightarrow$ NOT p OR NOT q
NOT (p OR q) $\leftrightarrow$ NOT p AND NOT q